How To Create an Excel Macro by Using Automation from Visual Basic .NET

This article was previously published under Q303871

Article ID : 303871
Last Review : July 14, 2004
Revision : 7.1

SUMMARY

This step-by-step article describes how to automate Microsoft Excel from Microsoft Visual Basic .NET to create a workbook that contains a new macro that is associated with a **CommandBar** button.

Steps to Create the Sample Visual Basic .NET Application

1. Start Microsoft Visual Basic .NET.
2. On the **File** menu, click **New**, and then click **Project**. Select **Windows Application** from the Visual Basic Projects types. Form1 is created by default.
3. Add references to **Microsoft Excel Object Library**, **Microsoft Office Object Library**, and **Microsoft Visual Basic for Applications Extensibility Library**. To do this, follow these steps:
   a. On the **Project** menu, click **Add Reference**.
   b. Click the **COM** tab, click **Microsoft Excel Object Library**, and then click **Select**.

      Note Microsoft Office 2003 includes Primary Interop Assemblies (PIAs). Microsoft Office XP does not include PIAs, but they may be downloaded. For additional information about Office XP PIAs, click the following article number to view the article in the Microsoft Knowledge Base:

      328912 (http://support.microsoft.com/kb/328912/) INFO: Microsoft Office XP PIAs Are Available for Download

   c. Select **Microsoft Visual Basic for Applications Extensibility Library**, and then click **Select**.
   d. Click **OK** in the **Add References** dialog box to accept your selections.
4. On the **View** menu, click **Toolbox** to display the Toolbox, and add a button to Form1.
5. Double-click **Button1**. The code window opens at the **onClick** event for **Button1**. Add the following line above Public Class Form1:

```
Imports Office = Microsoft.Office.Core
```

6. In the code window, add the following code:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
      System.EventArgs) Handles Button1.Click
   Dim oExcel As Excel.Application
   Dim oBook As Excel.Workbook
   Dim oModule As VBIDE.VBComponent
   Dim oCommandBar As Office.CommandBar
   Dim oCommandBarButton As Office.CommandBarControl
   Dim sCode As String

   ' Create an instance of Excel, and show it to the user.
   oExcel = New Excel.Application()

   ' Add a workbook.
   oBook = oExcel.Workbooks.Add

   ' Create a new VBA code module.
   oModule = oBook.VBProject.VBComponents.Add(VBIDE.vbext_ComponentType.vbext_ct_StdModule)

   sCode = "sub VBAMacro()" & vbCr & _
      "   msgbox ""VBA Macro called"" " & vbCr & _
      "end sub"

   ' Add the VBA macro to the new code module.
   oModule.CodeModule.AddFromString(sCode)
```

```
        Try
            ' Create a new toolbar, and show it to the user.
            oCommandBar = oExcel.CommandBars.Add("VBAMacroCommandBar")
            oCommandBar.Visible = True

            ' Create a new button on the toolbar.
            oCommandBarButton = oCommandBar.Controls.Add(Office.MsoControlType.msoControlButton)
            ' Assign a macro to the button.
            oCommandBarButton.OnAction = "VBAMacro"
            ' Set the caption of the button.
            oCommandBarButton.Caption = "Call VBAMacro"
            ' Set the icon on the button to a picture.
            oCommandBarButton.FaceId = 2151
        Catch exc As Exception
              MessageBox.Show("VBAMacroCommandBar already exists.", "Error")
        End Try

        oExcel.Visible = True
        ' Set the UserControl property so that Excel does not shut down.
        oExcel.UserControl = True

        ' Release the variables.
        oCommandBarButton = Nothing
        oCommandBar = Nothing
        oModule = Nothing
        oBook = Nothing
        oExcel = Nothing

        ' Force garbage collection.
        GC.Collect()

    End Sub
```

7. Add the following code to the top of Form1.vb:

```
Imports Office = Microsoft.Office.Core
Imports Microsoft.Office.Interop
Imports VBIDE = Microsoft.Vbe.Interop
```

8. Press F5 to build and then run the program.
9. Click **Button1** to start Excel, insert the Visual Basic for Applications (VBA) code, and then add a new **CommandBar** control. Click the button on the CommandBar to run the VBA macro.

Additional Notes for Office XP

Microsoft Office XP and Microsoft Office 2003 applications have a security option that allows programmatic access to the VBA object model. If this setting is **Off** (the default), you may receive an error when you run the sample code. For additional information about this setting and how you can correct the error, click the following article number to view the article in the Microsoft Knowledge Base:

282830 (http://support.microsoft.com/kb/282830/) PRB: Programmatic Access to Office XP VBA Project Is Denied

REFERENCES

For additional information, click the following article number to view the article in the Microsoft Knowledge Base:

194611 (http://support.microsoft.com/kb/194611/) Create and Call an Excel Macro Programmatically from VB

APPLIES TO

- Microsoft Visual Basic .NET 2003 Standard Edition
- Microsoft Visual Basic .NET 2002 Standard Edition
- Microsoft Office Excel 2003

- Microsoft Excel 2002 Standard Edition

Keywords: kbhowtomaster kbautomation kbpia KB303871